# Harmonizing tasks to human knowledge and capacities

Neerincx, Mark Antonius

# Chapter 2

# Cognitive Tasks in Man-Machine Systems

This chapter reflects a top-down approach to task analysis in man-machine design consisting of an analysis of, first, the primary cognitive task and, second, the interaction tasks. Theories and models of cognitive tasks are discussed which comprise, more or less, the overall task performance: the division of skill-, rule-, and knowledge-based behavior, the distinction of resources in human information processing, and the GOMS-based models of computer users' task. Hereafter, the mapping of users' goals and knowledge on the computer system is discussed. In general, current analyses of cognitive tasks differ enormously, because they center around specific purposes and are based on different theories. Specific parts of the analyses might be of help to answer the two research questions posed in chapter 1.

*Keywords:* task analysis, human information processing, expertise, task knowledge, GOMS, task-action mapping.

## 2.1 Introduction

The last chapter presented a brief outline of human-computer interaction (HCI) research. Much HCI-research focuses on the low-level interaction between user and computer aiming at efficient computer use and good human-machine dialogues. Examples are the modeling techniques employing a formal grammar to analyze and represent the knowledge the user needs to operate a user interface such as Task-Action Grammar (see, for example, de Haan *et al.*, 1991). However, problems of computer users to perform their tasks are not restricted to the usability and learn-ability of the computers. In chapter 1 it was already maintained that work changes as a consequence of the introduction of information technology. New problems may arise, such as diminishing flying skill of pilots or boredom of process operators. Harmonizing technology to the human does not only center around the interaction with a computer system, but should center around the work which is partly done with a computer. Consequently, theories or models of cognitive tasks are needed which are, more or less, able to describe the overall task of computer users. This chapter discusses some of such theories or models. For overviews of the HCI-literature we refer to Sutcliffe (1988), Johnson (1992), Dix *et al.* (1993), Eberts (1994), and Preece *et al.* (1994).

The term cognition is sometimes used in studies of artificial intelligence and multi-agent problem solving. In this thesis, cognition is used in a more restricted sense, that is, it is ascribed to a human individual (*cf.*, Gardner, 1987). In correspondence with this, cognitive tasks are

described in terms of the human information processes they involve. Before discussing more elaborate theories or models of cognitive tasks, we will first reiterate two important starting points of our study mentioned in Chapter 1.

The first one is that optimal human performance cannot be achieved solely through training. Many techniques for task analysis have been developed for training purposes. These techniques try to capture the knowledge and abilities needed for successful task performance. Tasks are decomposed into the psychological abilities that are required to perform the task and, subsequently, personnel is selected and trained on these abilities (Fleishman & Quaintance, 1984). This study pursues the opposite —possibly complementary— approach to bring about successful task performance by harmonizing the task to the person. The second starting point is that computer users are focussed on the execution of their tasks and do not want to invest in "secondary" activities such as consulting a manual or taking a course of training. Although such activities may be required for successful task performance, the computer users will hardly spent time on secondary activities and will mostly keep on trying to execute their primary task. The consequence of this narrow focus on the product of task execution is that the computers users fail in achieving this product. Carroll & Rosson (1987) called this the production paradox. Due to these two starting points, this chapter centers around modeling the "immediate" cognitive task performance leaving theories and analysis techniques for training of cognitive tasks out of consideration.

## 2.2 Skill-, Rule- and Knowledge-Based Behavior

Rasmussen (1983, 1984, 1986) developed a framework for cognitive task analysis in the domain of supervisory control of industrial installations. The analysis comprises a cognitive engineering approach to the design of supervisory control systems which is primarily directed at optimizing human diagnosis and minimizing human errors. In Rasmussen's framework, human behavior is a goal-oriented activity. For a main part based on "thinking-aloud" protocols, eight stages of decision making were identified: activation, observation, identification, interpretation, evaluation, goal selection, procedure selection, and execution. The stages resemble not a simple sequential ordering, but a step ladder in which stages can be by-passed through shortcuts. These shortcuts are developed as a result of training or experience. Figure 2.1 presents an overview of the step ladder model (the dashed arrows designate some possible shortcuts). This model is of special interest for the design of man-machine systems, because it describes observable behavior referring to covert, cognitive processes.

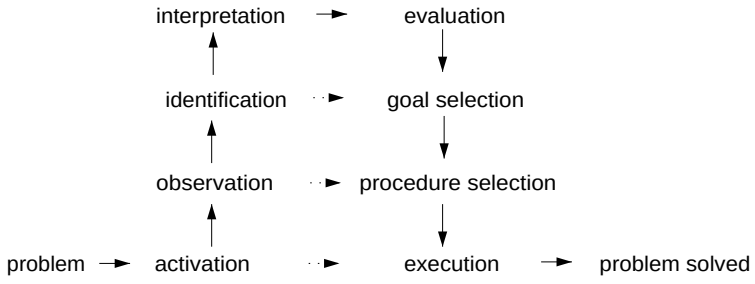


FIGURE 2.1: Rasmussen's (1984) step ladder model of decision making.

The step ladder comprises a three level hierarchy of the process control task: skill-based, rule-based and knowledge-based behavior. At the lower level actions are more specific and rigid,

while at the higher level actions are more general and flexible. Due to experience and training shortcuts are developed in the decision ladder model and, consequently, the tasks are performed at lower levels of information processing. It is worth noting that Hacker's action theory (1986) distinguishes three regulation levels with similar characteristics.

At the skill-based level, routine actions, after a statement of an intention, take place as smooth, automated and highly integrated patterns of behavior. The performance is characterized by feed-forward control. This type of behavior is initiated by signals (*i.e.,* time-space variables such as the set point of a flow-meter). An example of skill-based behavior is typing of skilled typists.

The rule-based level is applicable to tackling familiar problems. Solutions to these problems are governed by stored rules (productions) of the type "*if* state x *then* action y". These procedures may be learned explicitly through training, or more implicitly through experience. The performance is also characterized by feed-forward control. This type of behavior is initiated by signs indicating a state in the environment with reference for certain conventions for acts (*e.g.,* a flow-meter and valve can, jointly, be in several states invoking specific acts). For example, experienced users of a text editor achieve their cut and paste actions on the rule-based level.

The knowledge-based level comes into play in novel situations for which actions must be planned. Based on a mental model, the person sets local goals, initiates actions to achieve them, observes the extent to which the actions are successful and, if needed, poses new subgoals to minimize the discrepancy between the present position and the desired state. Control is primarily of the feed-back kind. This type of behavior is initiated by symbols (*i.e.,* abstract constructs defined by a formal structure of relations and processes such as a flow diagram comprising pipes, flow-meters and valves). Diagnosis of rarely occurring malfunctions in a process installation is an example of knowledge-based behavior.

Reason (1990) worked Rasmussen's skill-rule-knowledge classification of human performance out into the generic error-modeling system (GEMS). This system is a conceptual framework for the analysis of two error types: (i) slips and lapses (*i.e.,* where the actions do not go according to plan) and (ii) mistakes (*i.e.,* where the plan itself is inadequate to achieve its objectives). In Figure 2.2 the dynamics of GEMS are outlined. It permits identification of three basic error types: skill-based slips and lapses, rule-based mistakes and knowledge-based mistakes.

Skill-based slips and lapses occur prior to problem detection. They are mainly associated with *monitoring* failures. Attentional checks are required for good task performance. Errors are caused by inattention (*i.e.,* the omission of such checks) or over-attention (*i.e.,* mistiming of checks).

Mistakes are associated with problem solving. Problem solvers first establish whether or not the local indications have been encountered before. If the indications are recognized (at large part automatically) a previously established if (condition) then (actions) rule-based solution is applied. A hierarchy of rules exist: whenever exceptions are encountered, increasingly more specific rules are created at the lower levels of the hierarchy. Failures are due to misapplications of good rules or application of bad rules.

Only when the relatively effortless activity of applying existing rules fails to provide an adequate solution will problem solvers move to the more laborious mode of making inferences from current knowledge (mental models) and, from this, go on to formulate and try out various remedial possibilities. Errors may occur as a consequence of "bounded rationality" or an incomplete or inaccurate mental model of the problems space. "Bounded rationality" refers to human capacity for formulating and solving complex problems which is very small compared with the size of the problems where solution is required for objectively rational behavior in the real world.

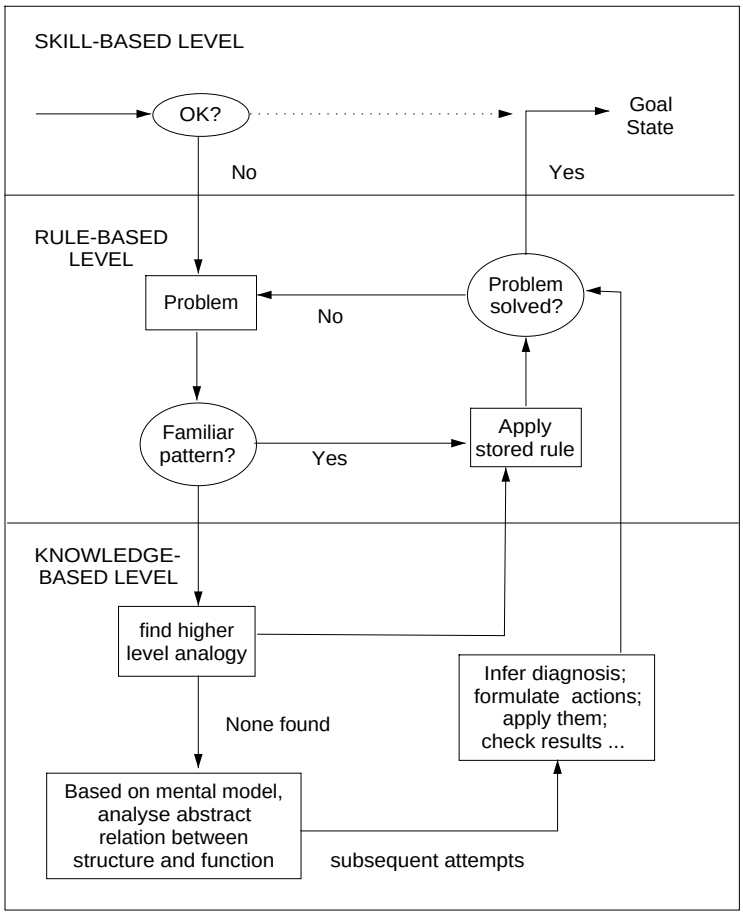


FIGURE 2.2: The cognitive processes in the Generic Error Modeling System of Reason (1990).

No matter how expert people are at coping with familiar problems, their performance will begin to approximate that of novices once their repertoire of rules has been exhausted by the demands of a novel situation. The important differences between the novice and expert are to be found at the skill-based and rule-based levels. Experts have a much larger collection of problem-solving rules than novices. They are also formulated at a more abstract level of presentation.

**Conclusions.** Rasmussen and Reason provide a framework distinguishing three levels of cognitive processes. The efficiency of these processes is for an important part determined by the task performers' experience and the task complexity. Higher levels of information processing demand more attention and call more on the limited capacity of working memory. Thus, the framework comprises a qualitative statement about cognitive task load, that is, load is high when the task performers have little experience and the task is complex. Task complexity refers to the learnability of routines which depends on the size of the problem space and the situational variability in problem solving. Chapter 3 will show that the skill-rule-knowledge (SRK) framework can be used to assess and design cognitive task load in man-machine systems.

Design of a software system should bring forth efficient human task performance (*cf.*, the action facilitation approach; Zijlstra, 1993; Arnold, 1993). According to the SRK-framework, task

performers are inclined to work at lower levels of information processing. For complex tasks, the application of procedural task knowledge at the rule-based level is efficient. Chapters 5 and 6 will go into a specific remedy for non-optimal problem solving of computer users: a computer function supplementing users' procedural task knowledge.

## 2.3 Multiple Resources in Information Processing

In the last section, behavior was described as a sequential process with specific information processing stages. In contrast to the "vertical" decision ladder model of the SRK-framework, multiple-resource theories comprise a "horizontal" model of information processing. According to these theories people have several different capacities with resource properties (Navon & Gopher, 1979; Wickens, 1984). Resources represent the mental effort supplied to improve the processing efficiency or the performance of a mental operation. Their deployment is under voluntary control and they are scarce (Wickens, 1992).

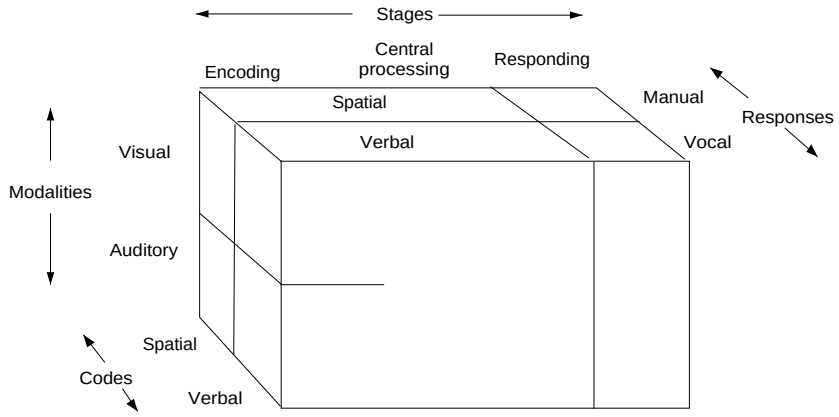


FIGURE 2.3: The multiple resource model of Wickens (1984).

Tasks, to be performed concurrently, will interfere more if more resources are shared. Wickens' model (1984) tries to account for the allocation of attention and the processing of information in complex tasks, such as piloting an aircraft, in which the operator works in a high-workload environment and has difficulty processing all the information. Figure 2.3 presents this model, in which resources are defined by three relatively simple dichotomous dimensions: two stage-defined resources (early versus late processes), two modality-defined resources (auditory versus visual encoding), and two resources defined by processing codes (spatial versus verbal).

In the *stage* dimension, the resources used for perceptual and central-processing activities appear to be the same, and they are functionally separate from those underlying the selection and execution of responses. The two perceptual *modalities* are distinguished to account for the possibility to divide attention between the eye and the ear, which is easier than between two auditory channels or two visual channels. The degree to which peripheral rather than central factors are responsible for better cross-modal time-sharing than intra-modal is uncertain. The *code*-related resource dimension may be defined, in part, by the two cerebral hemispheres: spatial processing taking place mainly in the right hemisphere and verbal processing mainly in the left hemisphere (Wickens, 1992). With respect to the codes for information processing, two principles of stimulus—central-processing—response-compatibility are conveyed. Verbal tasks seem to be

best served by auditory inputs and speech responses, whereas spatial tasks seem to be best served by visual inputs and manual responses (Wickens *et al.,* 1983; Wickens *et al.,* 1984).

**Conclusions.** The multiple-resource theory of Wickens sums up a lot of experimental data on dual-task performance. However, it does not encompass of all possible effects in dual task performances, such as switching, confusion and cooperation (Wickens, 1991). The interplay between cognitive processes proves to be rather complex and the "simple" attribution of cognitive tasks to a small number of resources addresses the interplay only partially. The cognitive neuroscience framework for multiple-task performance of Korteling (1994) seems to be better in addressing the interplay in multiple task performance. However, such a framework will not provide simple rules enabling one to predict multiple task performances. This point may reflect a fundamental problem, that is, the complexity of the cognitive processes in multiple task performance may not be covered by simple principles. Recently, next to the neuroscience framework, a modeling framework called EPIC is proposed to describe multiple-task performance. This EPIC-framework is discussed below as part of the GOMS-approach to model computer users' task performance.

## 2.4 GOMS-based Models of Cognitive Tasks

One of the most influential approaches to the study of the task of computer users is GOMS, which is an acronym of Goals, Operators, Methods and Selection rules (Card *et al.*, 1983). A GOMS-model describes the methods a computer user needs to accomplish his of her goals. A method is a series of steps consisting of operators that the user performs. If there are more methods to accomplish a goal, then the GOMS-model includes selection rules that choose the appropriate method depending on the context (see Figure 2.4).

```
GOAL: Edit-Manuscript
. GOAL: Edit-Unit-Task                  repeat until no more unit tasks
. . GOAL: Acquire-Unit-Task
. . . Get-Next-Page                     if at end of manuscript
. . . Get-Next-Task
. . GOAL: Execute-Unit-Task
. . . GOAL: Locate-Line
. . . . [select: Use-String-Search-Method
                 Use-Line-Feed-Method]
. . . GOAL: Modify-Text
. . . . [select: Use-S-Command
                 Use-M-Command]
. . . . Verify-Edit.
```

FIGURE 2.4: Example of GOMS-description (Card *et al.*, 1983).

The GOMS-framework is capable of producing a family of models at varying grains of analysis, because the level at which operators are defined may vary. There is no theoretical distinction between goals and operators (John *et al.*, 1990). The finer the grain of analysis, the more the operators are assumed to reflect basic psychological mechanisms. The coarser the grain of analysis, the more the operators reflect the specifics of the task environment.

The Model Human Processor (MHP) comprises the basic psychological mechanisms. It is based on the idea that results from cognitive (experimental) psychology can be cast in the form of a simplified model with parametric properties. Such a model can be useful in the design of systems. Card *et al.* (1983) developed the MHP to parameterize aspects of human information processing theories so that the execution time of tasks can be predicted. In this model, information is processed through three stages: the perceptual, cognitive, and motor stages.

GOMS and the corresponding MHP were first used to describe a rather simple task: the use of a text editor. Expert task performance was modeled leaving errors out of consideration. Below, some important elaborations, each trying to overcome a major limitation of these models, are discussed briefly. First, the Cognitive Complexity Theory was developed to enable predictions of learning times of software systems (Kieras & Polson, 1985). Second, SOAR is being developed to model, next to procedural task knowledge, human planning and problem-solving strategies (John *et al.*, 1990). Third, recently researchers are seeking to model multiple-task performance in a Executive Process-Interactive Control framework (Kieras & Meyer, 1994).

**The Cognitive Complexity Theory.** The Cognitive Complexity Theory (CCT) comprises an explicit model of the user's procedural knowledge entailed by a particular system design (Kieras & Polson, 1985). Based on the GOMS-framework and Anderson's (1987) learning theory, CCT is mainly used to describe and possibly predict the learning time to use a software system. According to CCT, problem-solving mechanisms attempt to achieve the learners goal in a novel task environment. Learning mechanisms then encode the results of successful problem-solving episodes as production rules that link the statement of a problem to its solution in a single processing step. If the need to solve the same problem occurs, the solution is provided by applying the applicable production rather than by repeating the earlier problem-solving episode (Polson, 1990). It should be noted that this learning model corresponds well with the skill-rule-knowledge framework described above. CCT may be viewed as a computational model of rule-based behavior which is the result of learning during problem solving. It should be noticed that the theory does not model this problem solving or knowledge-based behavior.

Bovair *et al.* (1990) showed that a production rule implementation of a GOMS-model can be used to make predictions about both learning and performance times of a editing task. In experiments of Vossen *et al.* (1987) there was a positive relationship between the number of new production rules as modeled by CCT and the learning times required for such a task.

**SOAR.** A major limitation of the first GOMS-model is that it only comprises expert's procedural task knowledge leaving human planning and problem solving strategies out of consideration. SOAR is an architecture for a system that is capable of general intelligence which is able to work on open-ended problems and employ problem-solving methods (Laird *et al.*, 1987). This architecture formulates all tasks in problem spaces, in which operators are selectively applied to the current state to attain desired states. Problem solving proceeds in a sequence of decision cycles that select problem spaces, states and operators, resulting in the application of the operator to move to a new state in the space.

Next to its objective of providing an architecture for general intelligence, SOAR is also intended as the basis for psychological theory. John *et al.* (1990) propose to use SOAR as architectural foundation upon which to build GOMS-models. Whereas GOMS can capture the procedural knowledge necessary to predict the course of behavior, SOAR allows GOMS to encompass the interruption

of tasks and responsiveness to the environment for browsing (Peck & John, 1992) and for an interactive computer game (John & Vera, 1992).

Another type of application of the SOAR implementation of problem spaces is the so-called programmable user model (PUM, Young & Whittington, 1990). The objective is to present the interface designer with a programmable cognitive architecture embodying psychological principles and constraints which he or she has to "instruct" to perform a range of representative tasks with a proposed interface. The instruction or PUM comprises the knowledge that would be necessary to accomplish these tasks. By executing the PUM, the stacking and un-stacking of problem spaces needed to accomplish the goal can be analyzed to measure the cognitive load of this task. This type of analysis can be used to predict task-action mapping errors in text-editor usage.

Next to SOAR, alternative cognitive architectures have been proposed. ACT-R is probably the most well-known alternative general cognitive architecture in which a cognitive skill is composed of production rules (Anderson, 1993). Another approach to cognitive simulation is to focus on modeling specific cognitive tasks such as fault management in process control (Roth *et al.* 1992; Furuta & Kondo, 1993) or acquisition of menu knowledge by exploration (Howes, 1994). An overview of the work on cognitive simulation is obviously beyond the scope of this study. Suffices it to say that much empirical research has to be conducted to test whether the simulations resemble human information processes and how far they can be used in practice.

**EPIC.** The Executive Process-Interactive Control (EPIC) framework comprises a human information-processing architecture that is especially suited for modeling multiple-task performance (Kieras & Meyer, 1994). This framework is being designed to couple the basic information processing and perceptual-motor mechanisms represented by production-system models such as CCT and SOAR. A recent re-examination of empirical results showed that the MHP is both incomplete and incorrect. Thus, the processors in EPIC are quite different from those in the MHP. The EPIC framework distinguishes a production and a long-term memory (*i.e.*, procedural and declarative knowledge) and the following seven processors: auditory, visual, cognitive (consisting of a production rule interpreter and working memory), vocal motor, manual motor, ocular motor, and tactile processor.

According to Kieras & Meyer, multiple processing threads can be represented simply as sets of rules that happen to run simultaneously. They hypothesize that all capacity limitations are the result of limited structural resources, rather than a limited cognitive processor (*cf.*, Wickens' multiple resource model, Figure 2.3). They also assume that certain apparent limitations in central capacity arise when modality-specific working memories must be used to maintain task information, but they have not yet tested this assumption in the EPIC framework. An executive control process, consisting of a set of production rules, coordinates the separate multiple tasks.

EPIC comprises GOMS-structured production rules *and* (concurrent) anticipatory actions (*e.g.*, eye movement) which may be based on a different goal. When the task is first learned, a GOMS-representation of the skill seems to be reasonable, in that the user's procedural knowledge is organized in terms of methods for accomplishing goals, and these methods are executed in a sequential hierarchical fashion. After extreme practice, the methods may become "flattened" and interleaved, so the original goal-hierarchical sequential form is lost in favor of a representation that permits the fastest possible temporal coordination of the processes with minimal cognitive activity (*cf.*, the skill-based level of Rasmussen).

**Examples of GOMS Applications.** An important motive to develop the GOMS-framework was to bridge to gap between cognitive theory and the practice of human-computer interaction (Card *et al.*, 1983). Consequently, the framework has been tested and used for real software applications, for example, to compare users' performance times of different workstations or to design knowledge provision.

*Comparing performance times.* GOMS has been applied to compare the work-times of telephone company toll and assistance operators on the current and a new workstation (Gray *et al.*, 1993). These operators have to determine who pays at what billing rate for a requested call. This is a rather simple task performed by experienced persons which are able to execute actions in parallel. To enable analysis of the task in terms of sequential and parallel MHP-processes, GOMS was extended with a critical path method (CPM, the critical path is the most time-consuming sequence of actions of the set of parallel action threads; *cf.*, the EPIC-framework). Subsequently, CPM-GOMS models were established based on, for the current workstation, observations of real operator task performances and, for the new workstation, system's specifications. Contrary to the expectations, the CPM-GOMS models predicted that task performance with the new workstation will be 3% slower then the current workstation. This prediction was confirmed in an empirical experiment. It is worth noting that this increase in performance time translates into a cost of almost $2 million a year to the telephone company.

*Designing knowledge provision.* Elkerton & Palmiter (1991) used the GOMS-framework to develop the structure and content of help information to be provided by a HyperCard system (a system to create and organize hyper-text information for later browsing and retrieval). They redesigned the existing help system and compared the resulting help with the original HyperCard help system. Users of the new help system proved to be faster in retrieving relevant information than users of the original system. The explicit procedural structure of the new help seemed to ease retrieval, whereas original system users had to learn the location of specific information.

Gong & Elkerton (1990) investigated whether the GOMS-framework can be useful to design manuals. They redesigned the original manual for a software system which provides information about occupational safety (*e.g.*, the likelihood of a lower back injury). An evaluation of the manuals with novices demonstrated better learning performances for the new manual.

Gugerty *et al.* (1991) report similar results. They investigated two methods to improve the organization and presentation of medical procedures on the computer: GOMS based versus medical-textbook based. Subjects who studied GOMS-based procedures remembered more about the procedures than the other subjects. Further, when asked to do a free recall of a procedure, subjects who had studied a textbook procedure often recalled key information in a location inconsistent with the procedure they actually studied, but consistent with the GOMS-based procedure. This result suggests that GOMS is a good model of procedural knowledge in human memory.

**Conclusions.** The GOMS-approach to task analysis seems to be very attractive: it centers around modeling human task performance and comprises both theoretical research on human information processing and applied research on human-computer interaction. The knowledge discerned about cognitive processes enables GOMS to predict performance times of simple, well-practiced tasks. A fundamental problem might be that for other tasks, new operators with specific parameters have to be hypothesized (*e.g.*, the MHP has been changed and extended in EPIC to model multiple-task performances and in SOAR to model problem-solving behavior). Currently, GOMS and its derivates might be better in modeling the task environment than modeling basic human information processes in complex, real-world tasks.

At this point, GOMS seems to be of special value: it can be viewed as a model of human procedural task knowledge and, consequently, be useful to design knowledge provision. The starting points are the goal-oriented nature of human behavior and the hierarchical organization of task knowledge. Knowledge provided in such a form seems to be easy to use and remember. Chapters 5 and 6 will present the design and evaluation of a computer function which presents this type of procedural task knowledge.

## 2.5 Mapping Users' Goals and Knowledge on the System

Above three major approaches towards the analysis of cognitive tasks were discussed: the SRK-framework, the multiple-resource theory and the GOMS-based models. Only GOMS provides an explicit, detailed analysis of the correspondence between the cognitive task and the operations of the software system. However, also GOMS insufficiently addresses the information transfer between user and system. In sum, the three approaches are lacking techniques to analyze and design the *interaction* between human and system, that is, the mapping of the users' task on the system's functionality and information presentation.

**Human-Computer Compatibility.** Norman (1986) maintains that there is a large misfit of the goals users have in mind to the operations the system can execute which results in non-optimal computer usage. In correspondence with Rasmussen, Norman describes the task of a computer user as a reactive, goal-directed process. Norman distinguishes seven stages of user activities involved in the performance of a task: establishing the goal, forming the intention, specifying the action sequence, executing the action, perceiving the system state, interpreting the state, and evaluating the system state with respect to the goals and intentions.

Norman distinguishes two gulfs between user and system: the gulfs of execution and evaluation. The gulf of execution refers to the problems users have in transferring their goals into commands the system "understands" and can execute, that is, in specifying the correct operations. The gulf of evaluation refers to the problems users have in understanding system behavior such as the interpretation of error messages. The correspondence between user's goals and system's operations and the correspondence between user's information needs and system's information provision can be analyzed at two levels: the functional and communication level (*cf.*, Moran, 1981; Frohlich & Luff, 1989; John & Vera, 1992).

*The functional level* comprises —in correspondence with GOMS— a hierarchical task decomposition. The terminal nodes of the hierarchy are the operations the system can perform, that is, the system's functions (*cf.*, the function-level operators of John *et al.*, 1990). With respect to the gulf of execution, the operations the computer can perform are often insufficiently compatible with the goals the user is willing to achieve and the order in which he or she wants to achieve them. For example, Neerincx (1990, 1992) analyzed the compatibility between users' task and two Computer Aided Design (CAD) systems. Both systems entailed incompatibilities: sometimes there was no direct mapping of the user's goals to systems' operations (*e.g.*, drawing a curve which touches two intersecting lines could not be done directly, but had to be done by drawing a circle, moving the circle and removing a part of the circle) and sometimes the sequence of user's goals during task execution did not correspond with the operations sequence the systems impose (*e.g.*, for projecting a point on an object, first, this object had to be selected and, second, the point had to be constructed, while the user wanted to do these two subtasks in the opposite

sequence). Similar evaluations with corresponding results were reported by Kieras & Polson (1985) and Sutcliffe & Springett (1992). With respect to the gulf of evaluation, the information provided by the system might not correspond to the information needs of the user. For the two above-mentioned CAD-systems the information provision proved to be insufficient, for example, the user interfaces did not provide information about the systems' modes.

Compatibility at the functional level is established by mapping user's goals —and corresponding goal sequences— on system's operations and mapping user's information needs on system's information provision. The mapping of user's goals on system's operations is called task-action mapping. According to Young (1983) with "task-action mapping" models the correspondence between the task domain and the action domain can be characterized. It suffices to represent the core-tasks and core-actions, so that you get the core of the mapping. The External-Internal Task (ETIT) analysis is a technique to model such mapping (Moran, 1983). An ETIT-analysis comprises an external and internal task space and mapping rules of the external on the internal task space. The external task refers to the user's goals (*e.g.*, move text) and corresponding concepts (*e.g.*, character, word and sentence). The internal task space represents the operations the computer can perform (*e.g.*, cut and paste) and corresponding concepts (*e.g.*, string). The mapping rules model how the users have to reformulate their goals and concepts in computer terms (*e.g.*, "move text" has to be reformulated in "cut string and paste string"). Although ETIT is mentioned in the literature many times, applications to real systems seem to be lacking (de Haan *et al.*, 1991).

*The communication level* describes the dialogue language. At this level, a system designer has to choose a type of dialogue, such as text or auditive dialogues (*cf.*, the keystroke-level operators of John *et al.*, 1990). In particular, the communication is simple for DM- (Direct Manipulation) or WIMP-interfaces (Windows, Icons, Mouse and Pull-down/pop-up menus). Learning how to control such interfaces is easy, because the icons, menus and windows provide information about the actions which can be done with the system (Hutchins *et al.*, 1986; Shneiderman, 1987; Coats & Vlaeminke, 1987). For example, a menu-item "mean data-set" informs the user that he or she can accomplish this specific goal with the system (functional level) and with minor experience the user even knows how to accomplish this goal (*i.e.*, simple communication: clicking on this menu-item with the mouse).

**Analysis of Task Knowledge.** The discussion above centered mainly around the analysis of current systems' usability. The task-action mapping technique can be used in system design, but it insufficiently addresses users' information needs. The method for the analysis of knowledge requirements for work with information technology known as Task Analysis for Knowledge Descriptions (TAKD) does address these needs. Recent versions of this method are being developed to complement the process of the design and evaluation of computer systems aiming at systems that will be compatible with users' task knowledge (Johnson, 1992).

TAKD analyses tasks in terms of actions and objects, and identifies generic actions and generic objects which are assumed to be at the level of description that makes them independent of the technology and tasks in which they have been observed. The basis of TAKD is that people structure their knowledge in a particular way and that system design is supported by analyzing current task knowledge (Johnson, 1989). Task knowledge is assumed to include knowledge of goals, procedures, actions and objects (Johnson, 1992).

Task knowledge structures represent the knowledge people possess about the tasks they have previously learned and performed in a given domain. An analysis of these structures identifies representative and central aspects of the task knowledge and models people's knowledge in

terms of goal structures, procedural substructure and taxonomic substructure. The goal structure encompasses two general forms of relation between different goal states: hierarchical and control relations. The procedural substructure reflects the detailed executable form of the task (at the lowest goal level). The taxonomic substructure represents the structure of objects and their attributes and features (Johnson, 1992).

Johnson (1992) briefly describes an applications of the task analysis for the design of a computer-assisted jewellery-design system. In this small case study, data collection techniques, such as structured interviews, provided data which were analyzed in terms of Task Knowledge Structure elements. A second stage of analysis was then performed to identify generalized Task Knowledge Structure elements that are common to many tasks and/or many task performances. A generic object is, for example, "shape" of which "round" is a typical specific object.

**Conclusions.** This section provided a short discussion of the main concepts involved in the analysis of human-computer compatibility. Much research has been conducted on this issue, but current knowledge is rather scrappy and current design techniques comprise compatibility only partially. The discussion did convey a top-down approach to the design of compatibility. First, the user's task has to be decomposed resulting in the establishment of system's functions. Task-action mapping helps to establish system operations fitting user's goals. Second, a user interface style has to be chosen which fits the future users. In general, a WIMP-interface seems to be the best choice for novices.

The TAKD-approach comprises a complete procedure for task analysis and provides, among other things, techniques for data collection. The knowledge analysis of tasks contributes to design-idea generation: to identify the functionality of the system, the dialogue requirements, and presentation characteristics of the user interface. The potential of this method is embedded in the design of computer systems which are compatible with users' task knowledge. Establishing such a compatibility will be a major improvement to the current software design practice. However, this does not guarantee that the users will be capable of executing their tasks. Human limitations are insufficiently addressed in the TAKD-approach and important design issues are left out of consideration, such as task allocation and human-computer cooperative task performance.

Taken together, there is no compound design technique which ensures a fine mapping of the cognitive tasks on the system's functionality and information presentation. This issue is pursued in Chapter 5 which presents a framework for model-based design of highly interactive computer systems incorporating task-action mapping and the analysis of information needs.

## 2.6 Discussion

This chapter discussed current approaches to the analysis of cognitive tasks which, more or less, comprise the complete task of a computer user and may be of use for the harmonization of tasks to human knowledge and capacities. One generic cognitive task analysis does not exist, because different purposes pose different requirements on such an analysis (*cf.*, Merkelbach & Schaagen, 1994). This chapter's discourse reflected a general top-down approach to task analysis in man-machine system design consisting of an analysis of, first, the overall or primary task independent of a specific technology and, second, the interaction tasks (this issue will be pursued in chapter 5). The specific ingredients and the depth of the analysis depend on the analysis' purpose.

Rasmussen and Reason developed their framework to improve the design of process control

systems. Although they were influenced by diverse theories, the main part of their work is founded on analyzing operators work in practice. GOMS is more thoroughly based on cognitive theory. However, the cognitive processes GOMS distinguishes are better in predicting execution times of simple, well-practiced tasks than for complex tasks. Cognitive simulation of human information processing is still in development and for complex, real-world tasks not yet very well possible.

Human cognition is for an important part purposeful, directed at achieving goals and removing obstacles to those goals (Card *et al.*, 1983, Rasmussen, 1986; Anderson, 1990). Next to simulation, GOMS can be used as an expert or prescriptive model for human goal attainment. Used this way, it supplements the skill-rule-knowledge framework, that is, SOAR models expert knowledge-based behavior, CCT models expert rule-based behavior, and EPIC models expert skill-based behavior. In particular, GOMS seems to be good at modeling the procedural task knowledge involved in rule-based behavior.

The GOMS task analysis starts where traditional, non-cognitive task analysis stops (Kieras, 1988). However, by far the most difficult problem is conducting the GOMS-based task analysis, that is, the acquisition of the information required for the construction of the GOMS-model. Current GOMS-analyses take user goals as given, the problem is to obtain the goals users will have, that is, the construction of the representation of the procedural knowledge that the user must have in order to operate the system (Bovair *et al.*, 1990). The TAKD-approach provides techniques to acquire task information from task performers. Further, the knowledge analysis in the TAKD-approach extends the knowledge structures considered in the GOMS-approach by including declarative knowledge about objects, their properties and the actions they can afford (*cf.*, Anderson, 1993).

In sum, the analyses of cognitive tasks discussed in this section center around different objectives. Specific parts of the analyses might be of help to answer the two research questions posed in chapter 1. In the next chapter, among other things, the skill-rule-knowledge framework is used to develop a method for the analysis and design of cognitive task load aiming at the allocation of a set of tasks to persons which correspond to their capacities. In the second part of this study, modeling procedural task knowledge in a GOMS-isomorph way and task-action mapping are, among other things, used to design an aiding function for the presentation of context-specific task knowledge.